

2/5/1

DIALOG(R) File 351:Derwent WPI
(c) 2005 Thomson Derwent. All rts. reserv.

012642597 **Image available**
WPI Acc No: 1999-448702/ 199938
XRPX Acc No: N99-335188

Programmable data processor for multimedia data containing e.g. still image, moving image, audio - has CPU that regulates programming unit in response to execution content of application program to update PLD

Patent Assignee: MATSUSHITA DENKI SANGYO KK (MATU)

Number of Countries: 001 Number of Patents: 001

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
JP 11184718	A	19990709	JP 97350981	A	19971219	199938 B

Priority Applications (No Type Date): JP 97350981 A 19971219

Patent Details:

Patent No	Kind	Lan Pg	Main IPC	Filing Notes
JP 11184718	A	11	G06F-009/46	

Abstract (Basic): JP 11184718 A

NOVELTY - The programmable data processor has a programmable logic device (PLD) (3) that can form a logic circuit by a program. A memory (2) stores the program data for the PLD. A programming unit (6) executes the program of the PLD using the stored program data. A CPU (1) regulates the programming unit in response to the execution content of the application program to update the PLD.

USE - For processing multimedia data containing e.g. still image, moving image, audio.

ADVANTAGE - Has improved speed and low cost since all-purpose PLD, which can flexibly correspond to whole digital process, is used. Performs logic circuit modification using other program data even in the middle of a process. Logic circuit of PLD can be dynamically changed to cooperate and execute a task for every unit. Has simple hardware component and selector that chooses program data based on usage situation shown on a table. DESCRIPTION OF DRAWING(S) - The figure shows the block diagram of the data processor. (1) CPU; (2) Memory; (3) PLD; (6) Programming unit.

Dwg.1/12

Title Terms: PROGRAM; DATA; PROCESSOR; DATA; CONTAIN; STILL; IMAGE; MOVE; IMAGE; AUDIO; CPU; REGULATE; PROGRAM; UNIT; RESPOND; EXECUTE; CONTENT; APPLY; PROGRAM; UPDATE

Derwent Class: T01; W02

International Patent Class (Main): G06F-009/46

International Patent Class (Additional): G06T-001/20; H04N-007/24

File Segment: EPI

SP9C

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開平11-184718

(43)公開日 平成11年(1999) 7 月 9 日

(51)Int.Cl.⁶

識別記号

F I

G 0 6 F 9/46

3 4 0

G 0 6 F 9/46

3 4 0 F

G 0 6 T 1/20

15/66

K

H 0 4 N 7/24

H 0 4 N 7/13

Z

審査請求 未請求 請求項の数7 O L (全 11 頁)

(21)出願番号

特願平9-350981

(22)出願日

平成9年(1997)12月19日

(71)出願人 000005821

松下電器産業株式会社

大阪府門真市大字門真1006番地

(72)発明者 高田 周一

大阪府門真市大字門真1006番地 松下電器
産業株式会社内

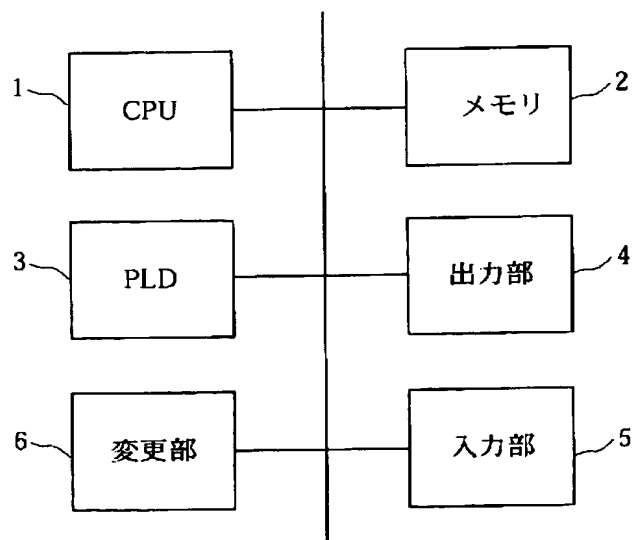
(74)代理人 弁理士 中島 司朗

(54)【発明の名称】 プログラマブルなデータ処理装置

(57)【要約】

【課題】 本発明は、あらゆる機能要求に対応しかつ低コスト化を図ったデータ処理装置を提供することを目的とする。

【解決手段】 メモリ2は、論理回路をプログラムにより形成可能なPLD3用の複数のプログラムデータを記憶する。変更部6は、プログラムデータを用いてプログラマブルロジックデバイスをプログラムする。CPU1は、アプリケーションプログラムの実行内容に応じて、プログラマブルロジックデバイスを更新するよう変更部6を制御する。



【特許請求の範囲】

【請求項1】 アプリケーションプログラムを実行するデータ処理装置であって、論理回路をプログラムにより形成可能なプログラマブルロジックデバイスと、プログラマブルロジックデバイス用の複数のプログラムデータを記憶する記憶手段と、プログラムデータを用いてプログラマブルロジックデバイスをプログラムするプログラミング手段と、前記アプリケーションプログラムの実行内容に応じて、プログラマブルロジックデバイスを更新するようプログラミング手段を制御する制御手段とを備えることを特徴とするデータ処理装置。

【請求項2】 前記データ処理装置は、さらにデータ退避領域を有する退避記憶手段と、プログラミング手段によるプログラミングの前に、プログラマブルロジックデバイス内の記憶素子のデータを退避記憶手段に退避させる退避手段と、プログラミング手段によるプログラミングの後に、当該プログラムデータに関する退避データを退避領域からプログラマブルロジックデバイスに復元する復元手段とを備えることを特徴とする請求項1記載のデータ処理装置。

【請求項3】 前記制御手段は、CPUのタスク切替えタイミングを検出する検出手段と、検出されたとき、切り替え後のタスクに対応するプログラムデータを選択する選択手段と、選択されたプログラムデータによりプログラマブルロジックデバイスを更新するようプログラミング手段に指示する指示手段とを備えることを特徴とする請求項1又は2記載のデータ処理装置。

【請求項4】 前記制御手段は、複数の逐次実行される複数のタスクからなるタスクウィンドウの切替えタイミングを検出する検出手段と、検出されたとき、切り替え後のタスクウィンドウに含まれるタスクに対応するプログラムデータを選択する選択手段と、選択されたプログラムデータによりプログラマブルロジックデバイスを更新するようプログラミング手段に指示する指示手段とを備えることを特徴とする請求項1又は2記載のデータ処理装置。

【請求項5】 前記制御手段は、CPUのタスク切替えタイミングを検出する検出手段と、検出されたとき、切り替え後に逐次実行される複数のタスクに対応する複数のプログラムデータを選択する選択手段と、選択された複数のプログラムデータによりプログラマブルロジックデバイスを更新するようプログラミング手段

に指示する指示手段とを備えることを特徴とする請求項1又は2記載のデータ処理装置。

【請求項6】 前記制御手段は、さらに、記憶手段に記憶されたプログラムデータの使用状況を示すテーブル手段を備え、

前記選択手段は、テーブル手段が示す使用状況に応じて複数のプログラムデータを選択することを特徴とする請求項5記載のデータ処理装置。

【請求項7】 前記記憶手段は、プログラマブルロジックデバイスにプロセッサとしても論理回路を形成するための特定のプログラムデータを記憶し、前記プログラム手段は、さらに、前記特定のプログラムデータを用いてプログラマブルロジックデバイスをプログラムすることを特徴とする請求項1又は2記載のデータ処理装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、コンピュータ、デジタルAV機器、デジタル通信装置などを構成するデータ処理装置に関する。

【0002】

【従来の技術】近年、パーソナルコンピュータ、デジタルAV機器、デジタル通信装置などのように、静止画、動画、音声などを含むマルチメディアデータを扱う種々のデータ処理装置が増加している。従来のデータ処理装置ではシステム全体の処理を加速させるため、CPUで処理していた特定の機能の一部をASIC (Application Specific Integrated circuit) などの専用ハードウェアで実行していた。特定の機能というのは、画像処理（グラフィックスデータの描画、動画像のデコード／エンコードなど）や信号処理（モデム機能、通信機能など）であり、大量の演算処理を必要とする。

【0003】従来のデータ処理装置がグラフィックス処理を加速する例を用いて説明する。図12は従来のデータ処理装置のブロック図を示す。同図において、101はCPU、102はメモリ、103はアクセラレータ、104は出力装置、105は入力装置である。メモリ102は、CPU101で実行するアプリケーションを格納している。このアプリケーションは、二次元（以下2D）あるいは三次元（以下3D）グラフィックス処理用であるものとする。また、その処理内容は、キーボードやマウスなどの入力装置105から入力される情報によって変化するものとする。例えばゲーム機のコントローラ入力によって表示内容が異なるなどの状況があげられる。処理の結果はコンピュータディスプレイなどの出力装置104を通して表示される。

【0004】アクセラレータ103は、2Dあるいは3Dグラフィックス処理に特化したパイプラインや並列処理機構あるいはDMA (Direct Memory Access) 機構を備える。このように構成されたデータ処理装置において、

3

CPU101は、2Dあるいは3Dグラフィックス処理の一部をアクセラレータ103に任せて処理を加速させる。

【0005】一般に、2Dあるいは3Dグラフィックス処理がバイト単位の処理の繰り返しとメモリ処理で占められ、CPU101にとってはバイト単位の処理の非効率さとメモリ処理の期間演算不能になることから高速化が難しい。また、CPU101とアクセラレータ103とが同時に処理すれば機能分担によりシステム全体が効率よく機能する。また、アクセラレータとして特化される機能は、上記の2Dあるいは3Dグラフィックス処理のほか、MPEGデコード・エンコードや、デジタル信号処理を要する通信機能など種々のものがある。以上のように従来のデータ処理装置は、専用機能に特化したアクセラレータを備えることにより、特定処理の加速を図っていた。

【0006】

【発明が解決しようとする課題】ところで、上記従来技術の構成では、アクセラレータがサポートする特定の処理しか加速されないので、マルチメディアデータ全般に対応して加速することができないという問題があった。

【0007】さらに、マルチメディアデータが浸透しつつある今日、マルチメディアデータの種類に応じて画像の圧縮／伸長、音声処理、グラフィックス処理、通信など様々な機能要求が増えてきている。これらの全部を加速するには、各々に別個のアクセラレータが必要になり、機能要求が増えるにつれハードウェア規模およびコストが増えることになる。

【0008】本発明は上記問題点を鑑み、あらゆる機能要求に対応しかつ低コスト化を図ったデータ処理装置を提供することを目的とする。

【0009】

【発明を解決するための手段】上記課題を解決するため本発明のデータ処理装置は、アプリケーションプログラムを実行するデータ処理装置であって、論理回路をプログラムにより形成可能なプログラマブルロジックデバイスと、プログラマブルロジックデバイス用の複数のプログラムデータを記憶する記憶手段と、プログラムデータを用いてプログラマブルロジックデバイスをプログラムするプログラミング手段と、前記アプリケーションプログラムの実行内容に応じて、プログラマブルロジックデバイスを更新するようプログラミング手段を制御する制御手段とを備えている。

【0010】また、前記データ処理装置は、さらにデータ退避領域を有する退避記憶手段と、プログラミング手段によるプログラミングの前に、プログラマブルロジックデバイス内の記憶素子のデータを退避記憶手段に退避させる退避手段と、プログラミング手段によるプログラミングの後に、当該プログラムデータに関する退避データを退避領域からプログラマブルロジックデバイスに復

4

元する復元手段とを備えて構成してもよい。前記制御手段は、CPUのタスク切替えタイミングを検出する検出手段と、検出されたとき、切り替え後のタスクに対応するプログラムデータを選択する選択手段と、選択されたプログラムデータによりプログラマブルロジックデバイスを更新するようプログラミング手段に指示する指示手段とを備える構成としてもよい。

【0011】

【発明の実施の形態】＜実施の形態1＞図1は、本発明の第1の実施形態におけるデータ処理装置のハードウェア構成を示すブロック図である。

【0012】このデータ処理装置は、CPU1、メモリ2、プログラマブルロジックデバイス（以下PLDと略す）3、出力装置4、入力装置5、変更部6からなり、PLD3を動的にプログラミングすることにより、PLD3を動的にプログラミングすることにより3Dグラフィックス描画やMPEGデコードなど各種のマルチメディアデータ処理を加速するように構成されている。PLD3は、内部の論理回路を外から与えられるプログラム情報に従って形成可能なプログラマブルロジックデバイスである。具体的には、PLD3は、内部にプログラム情報にしたがって配線可能な極めて多数の論理素子（ゲート）及び記憶素子（フリップフロップ、レジスタ、メモリ）と、与えられたプログラム情報によりそれらを配線するプログラミング回路とを有する。PLD3の一例としては、米国アルテラ社のFLEX 10KB ファミリ（ゲート数：10000～250000）などを用いることができる。これ以外のPLDでも動的にプログラミング可能なデバイスであればよい。

【0013】CPU1は、メモリ2に記憶されたOS（Operating System）プログラム及びアプリケーションプログラムを実行することに加えて、特定のプログラムを実行することにより、変更部6を通してPLD3を動的に変更（プログラミング）する。ここで特定のプログラムとは、タスクの切替え動作に合わせて、切替え後のタスクの処理内容に応じてPLD3を変更するためのプログラムである。このプログラムは、タスクの1つとしてもよいが、本実施形態ではOSの一部として実行されるものとする。例えば複数のアプリケーションを実行する場合、CPU1は、OSの下でアプリケーションをタスクとして時分割に切り替えて実行する。その際、CPU1は、上記特定のプログラムを実行することにより、タスクの切替えに併せてタスク毎に個別の処理を加速させるため、変更部6を通してPLD3を変更する。これにより、各タスクは、その処理内容に応じてPLD3により加速されることになる。

【0014】メモリ2は、OSプログラム及びアプリケーションプログラムの他に、上記の特定プログラムと、PLD3に対する複数のプログラム情報と、プログラム情報テーブルとを記憶する。図2はメモリ2の記憶内容

の具体例を示すメモリマップである。同図に示すようにメモリ2は、OS領域、タスクA領域、タスクB領域、プログラム情報領域を有する。

【0015】OS領域は、OSプログラムとタスク管理テーブルとを格納し、さらにOSデータエリアとからなる。タスクA領域は、タスクA（3Dグラフィックス処理用のアプリケーション）用の領域であり、タスクA

（プログラム本体とワークエリアを含む）を格納する領域21と、タスクAに関するPLD3内部データを退避するためのPLD退避領域22と、タスクAに関するCPU1内部のデータを退避するためのCPU退避領域23とを有する。タスクB領域も、タスクA領域と同様であるが、MPEGデコード処理用のアプリケーションである点異なる。

【0016】プログラム情報領域は、PLD3用のプログラム情報A（3Dグラフィックス処理用）と、プログラム情報B（MPEGデコード用）と、タスクとプログラム情報との対応関係を示すプログラム情報テーブルとを格納する。図3は、プログラム情報テーブルの具体例を示す図である。同図に示すように、プログラム情報テーブルは、タスクと、タスクが必要とするプログラム情報の種別とを対応させて記憶している。

【0017】図4は、CPU1によるPLD3の変更処理を示すフローチャートである。同図は、CPU1が上記の特定プログラムがOSの一部として実行される処理を示している。同図では、タスクA（3Dグラフィックス処理用）とタスクB（MPEGデコード処理用）との2つのタスクが交互に切り替えられながら処理される場合を示している。同図のようにCPU1は、一番最初に又はタスクBの次に実行されるタスクA（3Dグラフィックス処理）用の論理回路を形成するようにPLD3を変更（プログラミング）する（ステップ41）。その後OSによりタスクBへのタスク切替えがあれば（ステップ42：YES）、MPEGデコード処理用の論理回路を形成するようにPLD3を変更（プログラミング）する（ステップ43）。さらにOSにより3Dグラフィックスタスクへのタスク切替えがあれば（ステップ44：YES）、MPEGデコード処理用の論理回路を形成するようにPLD3を変更（プログラミング）する（ステップ41）。

【0018】このようにしてCPU1は、上記特定プログラムによりタスク切替えに際してPLD3を動的にプログラミングする。図5は、図4のステップ41に示した3Dグラフィックス処理への変更処理をより詳細に示すフローチャートである。

【0019】図5において、CPU1は、タスクB（MPEGデコード）からタスクA（3Dグラフィックス処理）へのタスク切替えの直前に、PLD3のMPEGデコード処理を停止し、CPU1内部のレジスタ情報を読み出してメモリ2上のCPU退避領域26に退避する

（ステップ51）。さらに、PLD3の内部情報（レジスタデータ、メモリデータなど記憶素子の記憶内容）をメモリ2上のPLD退避領域25に退避する（ステップ52）。これにより、停止したMPEGデコード処理に関するCPU1のレジスタ情報及びPLD3の内部情報が、図2に示したタスクB領域に格納される。さらに、CPU1は図3に示したプログラム情報テーブルを参照して、3Dグラフィックス用のプログラム情報がプログラム情報#1であることを特定して、そのプログラム情報#1を読み出して変更部6に供給する。これを受けて変更部6は、タスクA（3Dグラフィックス）用の論理回路を形成するようPLD3を変更（プログラミング）する（ステップ53）。これにより、PLD3は新たに3Dグラフィックス用の論理回路を形成する。

【0020】次いで、CPU1は、メモリ2のCPU退避領域23から3Dグラフィックス処理用に退避されていたレジスタ情報を読み出して、そのレジスタ情報をCPU1に復帰させ（ステップ54）、さらに、メモリ2のPLD退避領域22から3Dグラフィックス処理用に退避されていた内部情報を読み出して、その内部情報をPLD3に復帰させ（ステップ55）、PLD3の処理を再開（イネーブル）させる。これによりPLD3は、前回のタスク実行時に退避していた情報を復帰させ、再開することになる。図6は、図4のステップ43に示したMPEGデコード処理への変更処理をより詳細に示すフローチャートである。

【0021】図6において、CPU1は、タスクA（3Dグラフィックス処理）からタスクB（MPEGデコード）へのタスク切替えの直前に、PLD3の3Dグラフィックス処理を停止し、CPU1内部のレジスタ情報を読み出してメモリ2上のCPU退避領域23に退避する（ステップ61）。さらに、PLD3の内部情報をメモリ2上のPLD退避領域22に退避する（ステップ62）。これにより、停止した3Dグラフィックス処理に関するCPU1のレジスタ情報及びPLD3の内部情報が、図2に示したタスクA領域に格納される。さらに、CPU1は図3に示したプログラム情報テーブルを参照して、MPEG用のプログラム情報がプログラム情報#2であることを特定して、そのプログラム情報#2を読み出して変更部6に供給する。これを受けて変更部6は、タスクB（MPEGデコード）用の論理回路を形成するようPLD3を変更（プログラミング）する（ステップ63）。これにより、PLD3はMPEGデコード用の論理回路を形成する。

【0022】次いで、CPU1は、メモリ2のCPU退避領域26からMPEGデコードに退避されていたレジスタ情報を読み出して、そのレジスタ情報をCPU1に復帰させ（ステップ64）、さらに、メモリ2のPLD退避領域25からMPEGデコード処理用に退避されていた内部情報を読み出して、その内部情報をPLD3に

10

20

30

40

50

復帰させ(ステップ65)、PLD3の処理を再開する。これによりPLD3は、前回のタスク実行時に退避していた情報が復帰され、MPEGデコード処理を再開することになる。以上のように構成された本発明の第1実施形態におけるデータ処理装置について、その動作を説明する。

【0023】まず、本データ処理装置においてタスクB(MPEGデコード処理)からタスクB(3Dグラフィックス処理)にタスク切替えが生じた場合の動作を説明する。図7(a)は、図5に示したMPEGデコード処理から3Dグラフィックス処理への変更動作を説明する図である。同図(a)中の破線a1、a2に示すように、CPU1は、タスクBからタスクAへのタスク切替えの直前に、PLD3のMPEGデコード処理を停止し、CPU1内部のレジスタ情報、PLD3の内部情報をそれぞれCPU退避領域26、PLD退避領域25に退避する。

【0024】次に、CPU1は、3Dグラフィックス用のプログラム情報#1を読み出して変更部6を通して、タスクA(3Dグラフィックス)用の論理回路を形成するようPLD3を変更(プログラミング)する。らに、同図(a)中の破線a3、a4に示すように、CPU1は、メモリ2のCPU退避領域23のレジスタ情報、PLD退避領域22の内部情報を、それぞれCPU1、PLD3に復帰させる。PLD3の処理を再開(イネーブ)させる。これによりPLD3は、前回のタスク実行時に退避していた情報を復帰させ、再開することになる。

【0025】CPU1自身のタスクBへの切替え後は、同図(b)に示すように、PLD3は3Dグラフィックス処理用の論理回路を形成するので、CPU1によるタスクAの処理と協働して3Dグラフィックス処理を行う。図8は、PLD3がCPU1と協働して3Dグラフィックス処理を行う具体例を示す説明図である。

【0026】同図において、31はビデオRAM(フレームメモリ)に格納された3Dグラフィックスの描画画像を示す。32はメモリ2のタスクAのワークメモリに作成される3Dグラフィックス画像の頂点の座標テーブルを示す。33は、外部(例えば図外の磁気ディスク装置)から与えられる3Dグラフィックス画像のデータベースを示す。この場合CPU1とPLD3は、次のように機能分担する。すなわちCPU1は、外部からの描画すべき3Dグラフィックスデータを決定し、それを外部装置からタスクAのワークメモリ(3Dグラフィックス画像のデータベース33)に読みだし、さらに、3Dグラフィックスデータベースから3Dグラフィックス画像の各頂点の座標を算出して座標テーブルを作成し、各頂点で定まる多角形の塗り潰しをPLD3に指示する。

【0027】PLD3は、CPU1から塗り潰し指示を受けて座標テーブルを参照して、個々の多角形を塗り潰

す描画処理を行う。この描画処理は、いわゆるラスター変換やスキャンコンバージョン(座標データから表示データへ変換)、テクスチャ処理、ポリゴンの描画処理などである。描画処理では、大量の繰り返し演算を必要とするため、PLD3に形成された論理回路による加速の効果が大きい。さらに、図7(b)は、図6に示したMPEGデコード処理から3Dグラフィックス処理への変更動作を示す説明図である。同図(a)と同様に、CPU1は、タスク切替えに際して、破線c1、c2に示すように3Dグラフィックス処理用のレジスタ情報、内部情報の退避を行った後、MPEGデコード処理用のプログラム情報#2を用いて変更部6を通してPLD3をプログラミングし、破線c3、c4に示すようにMPEGデコード処理用のレジスタ情報、内部情報を復帰させる。

【0028】その後タスクBに切り替えられた後、CPU1とPLD3は協働してMPEGデコード処理を実行する。この場合の機能分担については、MPEGデコード処理に含まれる可変長符号の復号処理、逆離散余弦変換、逆量子化、動き補償処理のうち、定型的反復的な演算を要する逆離散余弦変換、逆量子化、動き補償処理又はその一部をPLD3が分担することが望ましい。これらの機能分担は、PLD3のゲート数及び処理能力と、CPU1の処理能力に応じて決定される。以下MPEGデコードが実行され、タスク切替えにより上記タスクA、Bが交互に繰り返される。タスクのバランスはアプリケーションに重みを指定してやることで、任意の比率のタスク切り替えが可能である。これはOSのタスク管理能力に依存する。

【0029】以上のようにPLD3はタイムシェアリングを行いながら汎用のアクセラレータとして使用されるため、プログラマブルでかつ高速かつ効率的なデータ処理が可能となる。なお、上記実施形態において、特定プログラムによるOSの一部の機能としてCPU1が、タスク切替えに際してPLD3の動作を停止し、プログラミング後に再開させていたが、アプリケーション(タスクA、B)が停止と再開をさせるようにしてもよい。これは、CPU1(タスク)とPLD3とはマスターとスレーブという協働関係にあるので、タスクによりPLD3の動作を停止/再開させることは容易に実現可能である。

【0030】＜実施の形態2＞本実施形態におけるデータ処理装置のハードウェア構成は、第1実施形態の図1と同じであるので説明を省略する。以下、CPU1が特定のプログラムを実行することにより実現されるPLD3の動的な変更処理について、第1実施形態と同じ点は説明を省略し、異なる点を中心に説明する。異なる点は、第1実施形態におけるデータ処理装置がタスクを切替える毎にPLD3を変更していたことに対して、本実施形態のデータ転送装置は、逐次実行される複数のタス

10

20

30

40

50

タスクをタスクウィンドウとして扱い、タスクウィンドウの切替える毎に、当該複数のタスクに対応するプログラム情報を用いてPLD3を変更するように構成している点である。さらに、本実施例では、CPU1は、PLD3をプログラムする際に、複数の別個のプログラム情報により形成される論理回路を併存させるように変更部6を通してプログラミングする点が異なっている。そのため、メモリ2に記憶されている上記特定プログラムの内容及びプログラム情報テーブルが異なっている。

【0031】以下、タスクウィンドウの具体例を用いて、本実施形態の特定プログラムにより実現されるCPU1の機能について説明する。メモリ2は、OSプログラム及びアプリケーションプログラムの他に、上記の特定プログラムと、PLD3に対する複数のプログラム情報と、タスクウィンドウとプログラム情報の対応関係を示したプログラム情報テーブルとを記憶する。図9はメモリ2の記憶内容の具体例を示すメモリマップである。

【0032】同図に示すようにメモリ2は、OS領域、タスクウィンドウ#1～#3、プログラム情報領域を有する。タスクウィンドウは、PLD3の変更なしで実行可能なタスクをグループ化したものである。同図において、タスクA、D、G、Hは、それぞれ3Dグラフィックス処理の一部を分担して実行する。これに伴い各タスクに対応するプログラム情報a～dが設けられている。以下、分担する各処理を3D#1～#4と呼ぶ。

【0033】また、タスクB、C、E、Fは、MPEGデコード処理の一部を分担して実行する。これに伴い各タスクに対応するプログラム情報e～hが設けられている。以下、分担する各処理をMPEG#1～#4と呼ぶ。タスクウィンドウ#1は、タスクA、B、Cの各領域からなる。各タスク領域は、タスクのプログラム本体（ワークエリアを含む）と、退避領域とからなる。退避領域は、第1実施形態のPLD退避領域とCPU退避領域とからなり、本実施例ではそれらの総称としている。タスクウィンドウ#1におけるタスクA、B、Cは、それぞれ3D#1、MPEG#2、MPEG#3を実行する。

【0034】同様にタスクウィンドウ#2は、タスクD、E、Fの各領域からなる。タスクD、E、Fは、それぞれ3D#2、MPEG#1、MPEG#4を実行する。タスクウィンドウ#3は、タスクG、Hの各領域からなる。タスクG、Hは、それぞれ3D#3、3D#4を実行する。

【0035】プログラム情報領域は、PLD3がタスクA～Hと協動して動作するためのプログラム情報a～hを記憶している。図10は、本実施例におけるプログラム情報テーブルの具体例を示す図である。同図に示すように、プログラム情報テーブルは、タスクウィンドウと、タスクウィンドウ内のタスクが必要とするプログラム情報の種別とを対応させて記憶している。

【0036】CPU1は、第1実施形態においてタスクの切替える毎にPLD3を変更していたことに対して、本実施形態では、タスクウィンドウの切替える毎にPLD3を変更するとして扱い、タスクウィンドウの切替える毎にPLD3を変更する点が異なっている。この動作は、図4に示したフローチャートにおいてステップ42の代わりに、タスクウィンドウの切替えを判断するステップを設けた構成となる点が異なる。ステップ44についても同様である。さらに、図5、6に示したタスク及びPLD3のプログラミングについては、CPU1は、CPU1のレジスタ情報及びPLD3の内部情報の退避と復帰（ステップ51、52、54、55、61、62、64、65）を、タスクウィンドウ内の複数のタスクについて行う点が異なる。さらにCPU1は、ステップ53、63において、プログラム情報テーブルに指定されている個々のプログラム情報の全てをPLD3にプログラムする必要があるため、CPU1は、当該個々のプログラム情報を編集して新たな1つのプログラム情報にしてから変更部6を通して変更する。

【0037】例えば、図10によればタスクウィンドウ#1に対応するのは、プログラム情報a、f、gの3つであり、他のタスクウィンドウからタスクウィンドウ#1への切替えに際してプログラム情報a、f、gを編集して新たなプログラム情報にする。具体的には、CPU1は、定められた記述言語により論理記述されている個々のプログラム情報をサブモジュールとして、サブモジュール全部を論理合成することにより新たなメインモジュールを作成する。この後、CPU1は作成されたメインモジュールを1つのプログラム情報として変更部6を通してPLD3をプログラミングする。上記の構成により、本実施形態のデータ処理装置は、タスクウィンドウ単位でPLD3が変更されることになる。その結果、タスク単位でPLD3が変更されるよりも、その変更時間を短縮することができる。

【0038】＜第3実施形態＞本実施形態におけるデータ処理装置のハードウェア構成は、第1、第2実施形態の図1と同じであるので説明を省略する。以下、CPU1が特定のプログラムを実行することにより実現されるPLD3の動的な変更処理について、第2実施形態と同じ点は説明を省略し、異なる点を中心に説明する。図9に示したメモリマップは、本実施例においても同じであるものとする。

【0039】異なる点は、本実施形態におけるデータ処理装置は、第2実施形態においてPLD3がタスクウィンドウの切替え毎に変更されていたのに対して、タスクの切替え毎に今後実行される複数のタスクに必要とされる複数のプログラム情報を用いて変更するように構成されている。そのため、メモリ2は、プログラム情報テーブルの代わりに、プログラム情報の使用状況を表すキャッシュテーブルが作成される。より詳しくいうと、CP

U 1 は、タスク切替えに際して、後に実行すべき複数のタスクを判定し、キャッシュテーブルを参照してそれらのタスクに必要なプログラム情報を選択し、それらのプログラム情報を上述したように論理合成して 1 つのプログラム情報を作成し、変更部 6 を通して P L D 3 をプログラミングする。

【0 0 4 0】図 1 1 にキャッシュテーブルの一例を示す。同図において、「プログラム情報種別」の欄は、図 9 に示したプログラム情報 a ~ h の種別を表す。「時刻」欄は、最後に C P U 1 により選択された時刻（従って最後にプログラミングされたときに選択された時刻）を表す。「サイズ」欄は、プログラム情報によりプログラムされた P L D 3 が占有されるサイズ（回路規模）を示す。サイズは、ゲート数や回路規模の割合でもよい。本実施例では説明の便宜上各プログラム情報が P L D 3 の 3 0 % を占有するものとする。「状態」欄は、現在 P L D 3 に論理回路が形成されている（o n）か否か（o f f）を表す。C P U 1 は、例えばこれから実行されるタスクを順に判定し、それらに対応するプログラム情報のサイズの合計が 1 0 0 %（あるいは所定のしきい値）を越えない範囲内で、選択候補とする。その際、C P U 1 は、状態が o n（現在 P L D 3 に論理回路が形成されている）のプログラム情報については、L R U（Least Recently Used）方式を用いて選択候補を決める。こうして選択候補となったプログラム情報を最終的に選択する。

【0 0 4 1】選択されたプログラム情報は、タスク切替えに際して P L D 3 にプログラミングされる。タスクの変更及びプログラミングの動作については、第 2 実施例と同様である。このとき、C P U 1 は、上記キャッシュ

テーブルを更新する。以上のように本実施形態によれば、タスク切替毎に、複数のプログラム情報を選択して P L D 3 を変更するので、タスクの生成、消滅が頻繁に生じる場合であっても、また、タスクスケジューリング（実行順序）が動的に変化する場合であっても、それぞれのタスク切替えに追従して適切なプログラム情報を P L D 3 にプログラムすることができる。

【0 0 4 2】その結果、C P U のキャッシュ動作と同じように P L D 3 の加速機能の使用効率が高くなり、プログラマブルでかつ高速かつ効率的なデータ処理が可能となる。なお、上記実施形態では、タスクの変化に応じて P L D 3 の論理回路が動的に変更されるが、タスクの中すなわちアプリケーションから P L D 3 のプログラムを変更するようにしてもよい。この場合、タスクの切り替えタイミングに関係なくアプリケーションから任意のタイミングで P L D 3 をプログラムすることになる。この結果、P L D 3 へのプログラムは静的なスケジューリングで実施される。従って、アプリケーションの処理の進行に従って、処理も加速されるので効率的である。また複数のアプリケーションが混在する場合、実施の形態 3

と 4 の動的スケジューリングと組み合わせれば効果を増す。

【0 0 4 3】また、上記各実施形態において、C P U 1 と変更部 6 とは、P L D 3 に組み込むようにしてもよい。この場合、図 1 から C P U 1 と変更部 6 を削った構成となる。この場合、P L D 3 は初期化時において、C P U 1 と同等の論理回路を形成するためのプログラム情報と、変更部 6 と同等の論理回路を形成するためのプログラム情報とを取り込んでプログラミングする初期化部を備えるように構成すればよい。この結果、簡単な構成で上記実施形態と同様に本発明をじっしすることができ。また、C P U 1 と変更部 6 の機能もプログラム情報が進化する度に機能向上する。さらに、上記第 2、第 3 実施形態では、C P U 1 が複数の個別プログラム情報を論理合成して新たな 1 つのプログラム情報を作成する例を示したが、P L D 3 自身が部分的にプログラム可能な構成であれば、論理合成しなくても個別プログラム情報を用いてプログラミングするようにしてもよい。

【0 0 4 4】また、上記各実施形態では、1 個の P L D を用いる例を示したが、複数の P L D を用いる構成としてもよい。

【発明の効果】

【0 0 4 5】本発明のデータ処理装置は、アプリケーションプログラムを実行するデータ処理装置であって、論理回路をプログラムにより形成可能なプログラマブルロジックデバイスと、プログラマブルロジックデバイス用の複数のプログラムデータを記憶する記憶手段と、プログラムデータを用いてプログラマブルロジックデバイスをプログラムするプログラミング手段と、前記アプリケーションプログラムの実行内容に応じて、プログラマブルロジックデバイスを更新するようプログラミング手段を制御する制御手段とを備えている。この構成によれば、汎用でかつ柔軟にデジタル処理全般に対応することができ、その結果、高速化と低コスト化を図ることができる。またプログラマブルロジックデバイスを用いているので、その交換で性能の向上が可能となるという有利な効果が得られる。

【0 0 4 6】また、前記データ処理装置は、さらにデータ退避領域を有する退避記憶手段と、プログラミング手段によるプログラミングの前に、プログラマブルロジックデバイス内の記憶素子のデータを退避記憶手段に退避させる退避手段と、プログラミング手段によるプログラミングの後に、当該プログラムデータに関する退避データを退避領域からプログラマブルロジックデバイスに復元する復元手段とを備えている。この構成によれば、上記効果に加えて、処理の途中であっても、他のプログラムデータを用いて論理回路変更することができる。

【0 0 4 7】さらに前記制御手段は、C P U のタスク切替えタイミングを検出する検出手段と、検出されたとき、切り替え後のタスクに対応するプログラムデータを

選択する選択手段と、選択されたプログラムデータによりプログラマブルロジックデバイスを更新するようプログラミング手段に指示する指示手段と備えている。この構成によれば、上記効果に加えて、タスクという単位毎に、タスクと協働して実行するようプログラマブルロジックデバイスの論理回路を動的に変更することができる。

【0048】また、前記制御手段は、複数の逐次実行される複数のタスクからなるタスクウィンドウの切替えタイミングを検出する検出手段と、検出されたとき、切り替え後のタスクウィンドウに含まれるタスクに対応するプログラムデータを選択する選択手段と、選択されたプログラムデータによりプログラマブルロジックデバイスを更新するようプログラミング手段に指示する指示手段とを備えている。この構成によれば、タスクウィンドウの切替え毎に、プログラマブルロジックデバイスを更新するので、更新のために発生する時間を少なくすることができる。また、前記制御手段は、CPUのタスク切替えタイミングを検出する検出手段と、検出されたとき、切り替え後に逐次実行される複数のタスクに対応する複数のプログラムデータを選択する選択手段と、選択された複数のプログラムデータによりプログラマブルロジックデバイスを更新するようプログラミング手段に指示する指示手段とを備えている。

【0049】この構成によれば、タスク切替毎に、複数のプログラムデータが選択されてプログラマブルロジックデバイスが変更されるので、タスクの生成、消滅が頻繁に生じる場合であっても、また、タスクスケジューリング（実行順序）が動的に変化する場合であっても、それぞれのタスク切替えに際して適切なプログラム情報をPLD3にプログラムすることができる。前記制御手段は、さらに、記憶手段に記憶されたプログラムデータの使用状況を示すテーブル手段を備え、前記選択手段は、テーブル手段が示す使用状況に応じて複数のプログラムデータを選択するよう構成される。

【0050】この構成によれば、タスク切替えに際して今後使用されるプログラム情報を効率良くPLD3にプログラムすることができる。また、前記記憶手段は、プ

【図3】

タスク	プログラム情報種別
A	#1
B	#2

ログラマブルロジックデバイスにプロセッサとしても論理回路を形成するための特定のプログラムデータを記憶し、前記プログラム手段は、さらに、前記特定のプログラムデータを用いてプログラマブルロジックデバイスをプログラムするよう構成されている。

【0051】この構成によれば、ハードウェア構成を簡単にすることができる。

【図面の簡単な説明】

【図1】本発明の実施例におけるデータ処理装置の構成を示すブロック図である。

【図2】メモリ2の記憶内容の具体例を示すメモリマップである。

【図3】プログラム情報テーブルの具体例を示す図である。

【図4】CPU1によるPLD3の変更処理を示すフローチャートである。

【図5】3Dグラフィックス処理への変更処理をより詳細に示すフローチャートである。

【図6】MPEGデコード処理への変更処理をより詳細に示すフローチャートである。

【図7】MPEGデコード処理と3Dグラフィックス処理との変更動作を説明図である。

【図8】PLD3がCPU1と協働して3Dグラフィックス処理を行う具体例を示す説明図である。

【図9】メモリ2の記憶内容の具体例を示すメモリマップである。

【図10】プログラム情報テーブルの具体例を示す図である。

【図11】キャッシュテーブルの一例を示す。

【図12】従来のデータ処理装置のブロック図を示す。

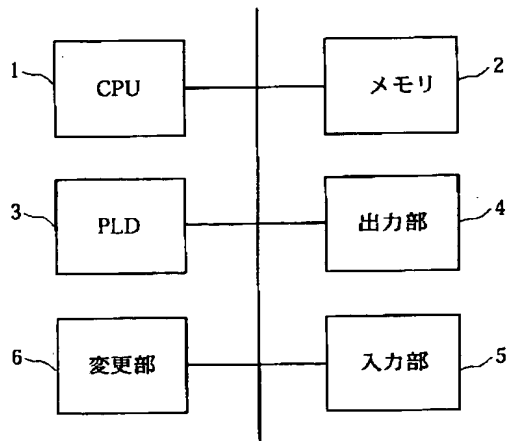
【符号の説明】

- 1 CPU
- 2 メモリ
- 3 PLD
- 4 出力装置
- 5 入力装置
- 6 変更部

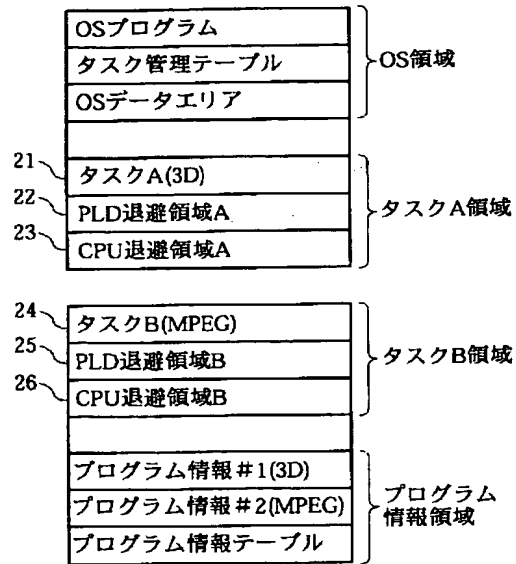
【図10】

タスクウィンドウ	プログラム情報種別
#1	a(3D#1).f(MPEG#2).g(MPEG#3)
#2	a(3D#2).e(MPEG#1).g(MPEG#4)
#3	a(3D#3).d(3D#4)

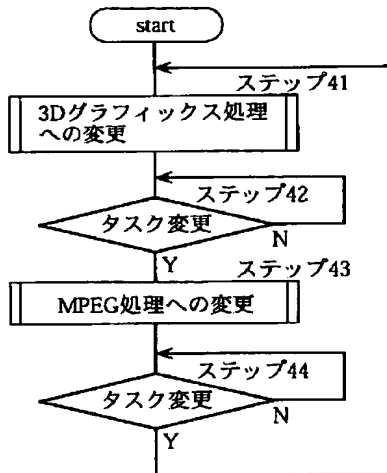
【図1】



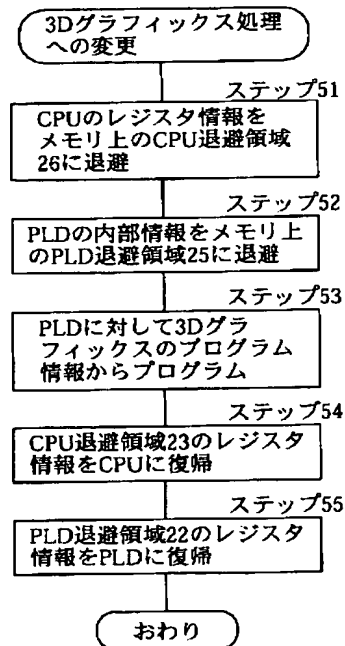
【図2】



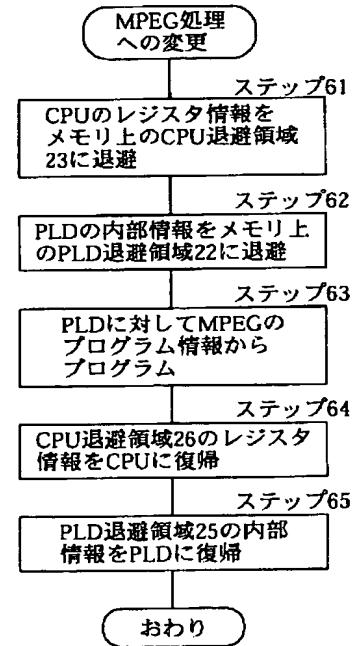
【図4】



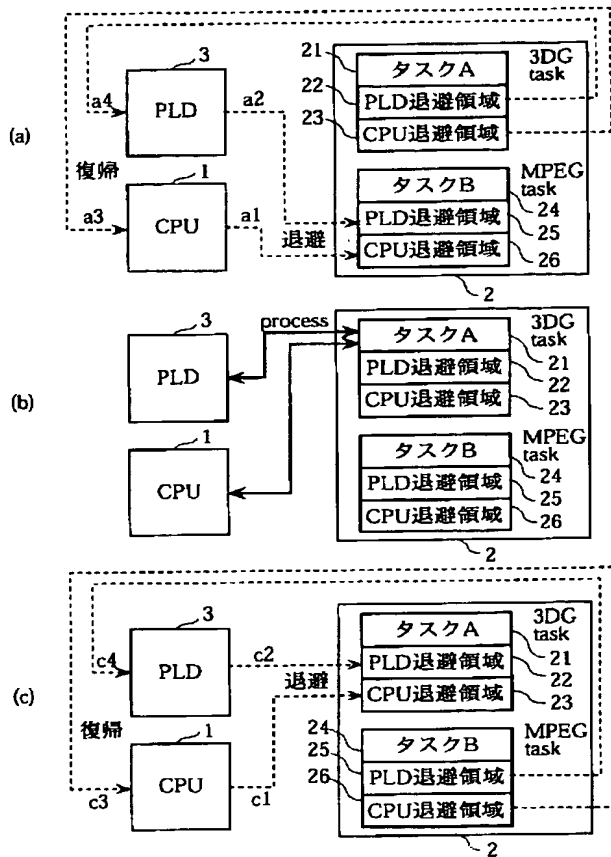
【図5】



【図6】



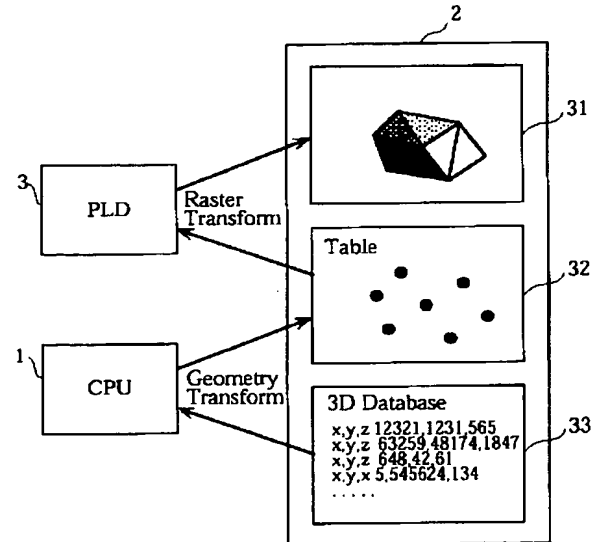
【図 7】



【図 1 1】

プログラム情報種別	時刻	サイズ	状態
プログラム情報a (3D#1)	XX:XX:XX	yyyy	on
プログラム情報b (3D#2)	XX:XX:XX	yyyy	off
プログラム情報c (3D#3)	XX:XX:XX	yyyy	off
プログラム情報d (3D#4)	XX:XX:XX	yyyy	on
プログラム情報e (MPEG#1)	XX:XX:XX	yyyy	off
プログラム情報f (MPEG#2)	XX:XX:XX	yyyy	off
プログラム情報g (MPEG#3)	XX:XX:XX	yyyy	off
プログラム情報h (MPEG#4)	XX:XX:XX	yyyy	on

【図 8】



【図 9】

OSプログラム	
タスク管理テーブル	
OSデータエリア	
タスクA(3D#1)	タスクウィンドウ#1
退避領域A	
タスクB(MPEG#2)	
退避領域B	
タスクC(MPEG#3)	タスクウィンドウ#2
退避領域C	
タスクD(3D#2)	
退避領域D	
タスクE(MPEG#1)	タスクウィンドウ#3
退避領域E	
タスクF(MPEG#4)	
退避領域F	
タスクG(3D#3)	タスクウィンドウ#4
退避領域G	
タスクH(3D#4)	
退避領域H	
プログラム情報a (3D#1)	プログラム情報領域
プログラム情報b (3D#2)	
プログラム情報c (3D#3)	
プログラム情報d (3D#4)	
プログラム情報e (MPEG#1)	
プログラム情報f (MPEG#2)	
プログラム情報g (MPEG#3)	
プログラム情報h (MPEG#4)	

【図 1 2】

